BigARTM: Open Source Library for Regularized Multimodal Topic Modeling of Large Collections

Konstantin Vorontsov[1,3,4], Oleksandr Frei[2], Murat Apishev[3], and Peter Romov[4]

[1] Department of Intelligent Systems at Dorodnicyn Computing Centre of RAS,
Moscow Institute of Physics and Technology, voron@forecsys.ru
[2] Schlumberger ...
[3] Moscow State University, ...
[4] Yandex ...

Abstract. Probabilistic topic modeling of text collections is a powerful tool for statistical text analysis. In this paper we announce the BigARTM open source project for regularized multimodal topic modeling of large collections. Several experiments on Wikipedia corpus shows that BigARTM performs faster and gives better perplexity comparing to other popular packages,such as Vowpal Wabbit LDA and Gensim. We also demonstrate several unique BigARTM features, such as topics regularization and multi-language models, which are not available in other software packages for topic modeling.

Keywords: probabilistic topic model, Probabilistic Latent Sematic Analysis, Latent Dirichlet Allocation, Additive Regularization of Topic Models, stochastic matrix factorization, EM-algorithm, BigARTM.

1 Introduction

Topic modeling is a rapidly developing branch of statistical text analysis [1]. Topic model reveals a hidden thematic structure of the text collection and finds a compressed representation of each document by a set of its topics. Such models appear to be highly useful for many applications including information retrieval for long-text queries, revealing research trends and research fronts, classification, categorization, summarization and segmentation of texts. More ideas, models and applications are outlined in the survey [4].

From the statistical point of view, a probabilistic topic model defines each topic by a multinomial distribution over words, and then describes each document with a multinomial distribution over topics. From the optimizational point of view, topic modeling can be considered as a special case of constrained approximate stochastic matrix factorization. Learning a factorized representation of a text collection is an ill-posed problem, which has an infinite set of solutions. Then a regularization must be used to impose problem-oriented restrictions and thus to choose a more reasonable solution.

Hundreds of models adapted to different situations are described in the literature. For practitioners, most of them are too difficult to quickly understand, compare, combine and embed into applications. Then there was a common practice to use only the simplest models such as *Probabilistic Latent Semantic Analysis*, PLSA [6] and *Latent Dirichlet Allocation*, LDA [3]. Some of the difficulties are rooted in Bayesian learning framework, which is dominating approach in topic modeling. Bayesian inference of each kind of model is a unique and laborious mathematical work, which prevents the unification and the flexible manipulation of various topic models. Until now, there was no freely available development tools to easily design, modify, select, and combine topic models.

In this paper we announce **the BigARTM open source project** for parallel distributed online topic modeling of large collections, `http://bigartm.org`. The theory of BigARTM is based on non-Bayesian multicriteria approach — *Additive Regularization of Topic Models*, ARTM [9]. In ARTM a topic model is learned by maximizing the weighted sum of the log-likelihood and additional regularization criteria. The optimization problem is solved by a general regularized expectation-maximization (EM) algorithm, in which any regularization criteria or their combination can be substituted. Many known Bayesian topic models were revisited in terms of ARTM in [11,10]. Compared to the Bayesian approach, ARTM makes topic models easier to design, infer, combine and explain, thus reducing barriers to entry into topic modeling research field.

BigARTM is released under New BSD License, which permits free commercial and non-commercial usage. The core of the library is written C++ and exposed via two equally rich APIs for C++ and Python. The library is cross-platform and can be built for Linux, Windows and OS X in both 32 and 64 bit configuration. In our experiments on Wikipedia corpus BigARTM performs better than Vowpal Wabbit LDA and Gensim libraries in terms of perplexity and runtime. Comparing to other libraries BigARTM offers several additional features, such as regularization and multi-modal topic modeling.

The rest of the paper is organized as follows. In section 2 we introduce a multimodal topic model generalized for documents with multiple types of accompanying discrete metadata. In section 3 we describe a fast online algorithm [5] generalized for any additively regularized topic models. In section 4 we consider the parallel architecture and implementation details of BigARTM library. In section 5 we report some experiments on large datasets. In section 6 we discuss advantages, limitations and open problems of BigARTM.

2 Multimodal regularized topic model

Let D denote a finite set (collection) of texts and W^1 denote a finite set (vocabulary) of all terms from these texts. Each term can be a single word or a key phrase. A document can contain not only words, but also terms of other modalities. Each modality is defined by the finite set (vocabulary) of terms W^m, $m = 1, \dots, M$. Examples of not-word modalities are: authors, class or category labels, date-time stamps, references to/from other documents, entities mentioned in texts, objects found in the images illustrating texts, users that read or downloaded documents, advertising banners, etc.

Assume that each term occurrence in each document refers to some latent topic from a finite set of topics T. Text collection is considered to be a sample of triples $(w_i, d_i, t_i),\ i = 1, \dots, n$, drawn independently from a discrete distribution $p(w, d, t)$ over a finite space $W \times D \times T$, where $W = W^1 \sqcup \dots \sqcup W^m$ is disjoint union of all modality vocabularies. Terms w_i and documents d_i are observable variables, while topics t_i are latent variables.

Following the idea of Correspondence LDA [2] and Dependency LDA [8] we introduce the topic model for each modality:

$$p(w \mid d) = \sum_{t \in T} p(w \mid t)\, p(t \mid d) = \sum_{t \in T} \phi_{wt}\theta_{td}, \quad d \in D,\ w \in W^m,\ m = 1, \dots, M.$$

The model parameters $\phi_{wt} = p(w \mid t)$ and $\theta_{td} = p(t \mid d)$ form the matrices $\varPhi^m = \bigl(\phi_{wt}\bigr)_{W^m \times T}$ of *term probabilities for the topics*, and $\varTheta = \bigl(\theta_{td}\bigr)_{T \times D}$ of *topic probabilities for the documents*. Matrices $\varPhi^m$, if stacked vertically, form $W{\times}T$-matrix $\varPhi$. Matrices $\varPhi^m$ and $\varTheta$ are *stochastic*, that is, their vector-columns represent discrete distributions. The number of topics $|T|$ is usually assumed to be much smaller than $|D|$ and $|W|$.

To learn parameters $\varPhi^m$, $\varTheta$ from the multimodal text collection we maximize the log-likelihood for each m-th modality:

$$\mathscr{L}_m(\varPhi^m, \varTheta) = \sum_{d \in D} \sum_{w \in W^m} n_{dw} \ln p(w \mid d) \to \max_{\varPhi^m, \varTheta},$$

where n_{dw} is the number of occurrences of the term $w \in W^m$ in the document d. Note that topic distributions of documents $\varTheta$ are common for all modalities. Following the ARTM approach, we add a regularization penalty term $R(\varPhi, \varTheta)$ and solve the constrained multicriteria optimization problem via scalarization:

$$\sum_{m=1}^{M} \tau_m \mathscr{L}_m(\varPhi^m, \varTheta) + R(\varPhi, \varTheta) \to \max_{\varPhi, \varTheta}; \tag{1}$$

$$\sum_{w \in W^m} \phi_{wt} = 1,\ \phi_{wt} \geq 0; \qquad \sum_{t \in T} \theta_{td} = 1,\ \theta_{td} \geq 0. \tag{2}$$

The local maximum $(\varPhi, \varTheta)$ of the problem (1), (2) satisfies the system of equations with auxiliary variables $p_{tdw} = p(t \mid d, w)$:

$$p_{tdw} = \operatorname*{norm}_{t \in T}\bigl(\phi_{wt}\theta_{td}\bigr); \tag{3}$$

$$\phi_{wt} = \operatorname*{norm}_{w \in W^m}\left(n_{wt} + \phi_{wt}\frac{\partial R}{\partial \phi_{wt}}\right); \quad n_{wt} = \sum_{d \in D} n_{dw} p_{tdw}; \tag{4}$$

$$\theta_{td} = \operatorname*{norm}_{t \in T}\left(n_{td} + \theta_{td}\frac{\partial R}{\partial \theta_{td}}\right); \quad n_{td} = \sum_{w \in d} \tau_{m(w)} n_{dw} p_{tdw}; \tag{5}$$

where $\operatorname*{norm}_{t\in T} x_t = \frac{\max\{x_t,0\}}{\sum\limits_{s\in T}\max\{x_s,0\}}$ transforms a vector $(x_t)_{t\in T}$ to a discrete distribution; $m(w)$ is the modality of the term w, so that $w \in W^{m(w)}$.

The system of equations (3)–(5) follows from Karush–Kuhn–Tucker conditions (see Appendix for the proof). It can be solved by various numerical methods. Particularly, the simple-iteration method is equivalent to the EM algorithm, which is typically used in practice. For single modality ($M = 1$) Theorem 1 gives the regularized EM algorithm proposed in [9]. With no regularization ($R = 0$) it corresponds to PLSA [6]. Many Bayesian topic models can be considered alternatively as special cases of ARTM with different regularizers R, as shown in [11,10]. For example, LDA [3] corresponds to the entropy smoothing regularizer.

Due to models unification and additive regularization BigARTM enables to build topic models for various applications simply by choosing a suitable combination of regularizers from a constantly growing user-extensible library.

3 Online topic modeling

Following the idea of Online LDA [5] we split the collection D into batches D_b, $b = 1, \dots, B$, and organize EM iterations so that each document vector θ_d is iterated until convergence at a constant matrix $\mathit{\Phi}$, see Algorithm 1 and 2. Matrix $\mathit{\Phi}$ is updated rarely, after all documents from the batch are processed. For a large collection, matrix $\mathit{\Phi}$ stabilizes after a small initial part of the collection passed. Therefore a single pass through the collection is usually sufficient to learn a topic model. The second pass may be needed for the initial part of the collection.

Algorithm 1 does not specify how often to synchronize $\mathit{\Phi}$ matrix at step 5. It can be done after every batch or less frequently (for instance if $\frac{\partial R}{\partial \phi_{wt}}$ takes long time to evaluate). This flexibility is especially important for concurrent implementation of the algorithm, where multiple batches are processed in parallel. In this case synchronization can be triggered when a fixed number of documents had been processed since last synchronization.

The online reorganization of the EM iterations is not necessarily associated with Bayesian inference used in [5]. Different topic models, from PLSA to multimodal and regularized models, can be learned by the above online EM algorithm.

4 BigARTM architecture

The main goal for BigARTM architecture is to ensure constant memory usage regardless of the size of the collection. For this reason each D_b batch should be stored on disk in a separate file, and only a limited number of batches will be loaded into the main memory at any given time. The entire $\mathit{\Theta}$ matrix is also never stored in the memory. As the result, the memory usage stays constant regardless of the size of the collection.

Algorithm 1 Online EM-algorithm for multimodal ARTM

```
Require: collection $D_b$, discounting factor $\rho \in (0, 1]$;
Ensure: matrix $\Phi$;
initialize $\phi_{wt}$ for all $w \in W$ and $t \in T$;
$n_{wt} := 0$,  $\tilde{n}_{wt} := 0$;
for all batches $D_b$, $b = 1, \dots, B$ do
    $\tilde{n}_{wt} := \tilde{n}_{wt} + \mathsf{ProcessBatch}(D_b, \phi_{wt})$;
    if (synchronize) then
        $n_{wt} := \rho n_{wt} + \tilde{n}_{dw}$ for all $w \in W$ and $t \in T$;
        $\phi_{wt} := \operatorname*{norm}_{w \in W^m} \bigl(n_{wt} + \phi_{wt} \frac{\partial R}{\partial \phi_{wt}}\bigr)$ for all $w \in W^m$, $m = 1, \dots, M$ and $t \in T$;
        $\tilde{n}_{wt} := 0$ for all $w \in W$ and $t \in T$;
```

Algorithm 2 $\mathsf{ProcessBatch}(D_b, \phi_{wt})$

```
Require: batch $D_b$, matrix $\phi_{wt}$;
Ensure: matrix $\tilde{n}_{wt}$;
$\tilde{n}_{wt} := 0$ for all $w \in W$ and $t \in T$;
for all $d \in D_b$ do
    initialize $\theta_{td} := \frac{1}{|T|}$ for all $t \in T$;
    repeat
        $p_{tdw} := \operatorname*{norm}_{t \in T} \bigl(\phi_{wt}\theta_{td}\bigr)$ for all $t \in T$;
        $n_{td} := \sum_{w \in d} \tau_{m(w)} n_{dw} p_{tdw}$ for all $t \in T$;
        $\theta_{td} := \operatorname*{norm}_{t \in T} \bigl(n_{td} + \theta_{td} \frac{\partial R}{\partial \theta_{td}}\bigr)$ for all $t \in T$;
    until $\theta_d$ converges;
    increment $\tilde{n}_{wt}$ by $n_{dw} p_{tdw}$ for all $w \in d$ and $t \in T$;
```

Concurrency. An general rule of concurrency design is to express parallelism at the highest possible level. For this reason BigARTM goes for concurrent processing of the batches and keeps a single-threaded code for the $\mathsf{ProcessBatch}(D_b, \phi_{wt})$ routine.

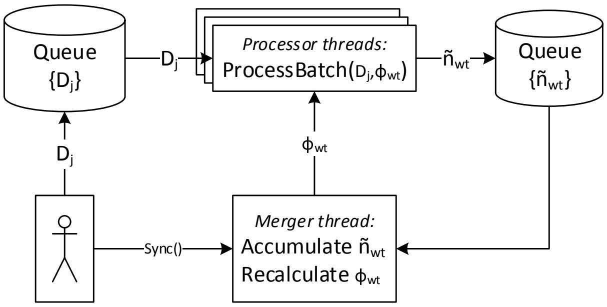

Fig. 1. Diagram of key BigARTM components

To run multiple ProcessBatch in parallel the inputs and outputs of this routine are stored in two separate in-memory queues, locked for push and pop operations with spin locks. This approach does not add any noticeable synchronization overhead because both queues only store smart pointers to the actual data objects, so push and pop operations does not involve copying or relocating big objects in the memory.

Smart pointers are also essential for handling of the ϕ_{wt} matrix. This matrix is *read* by all processors threads, and can be *written* at any time by the merger thread. To resolve this conflict we keep two copies of the ϕ_{wt} — an *active* Φ and a *background* Φ matrices. The active matrix is read-only, and is used by the processor threads. The background matrix is being built in a background by the merger thread at steps 6 and 7 of Algorithm 1, and once it is ready merger thread marks it as active. Before processing a new batch the processor thread gets the current active matrix from the merger thread. This object is passed via shared smart pointer to ensure that processor thread can keep ownership of the ϕ_{wt} matrix until the batch is fully processed.

Note that all processor threads share the same Φ matrix, which means that memory usage stays at constant level regardless of how many cores are used for computation. Using memory for two copies of the Φ matrix in our opinion gives a reasonable usage balance between memory and CPU resources. An alternative solution with only one Φ matrix is also possible, but it would require a heavy usage of atomic CPU instructions. Such operations are very efficient, but still come at a considerable synchronization cost[5], and using them for all reads and writes of the Φ matrix would cause a significant performance degradation for merger and processor threads. Besides, an arbitrary overlap between reads and writes of the Φ matrix eliminates any possibility of producing a deterministic result. The design with two copies of the Φ matrix gives much more control over this and in certain cases allows BigARTM to behave in a fully deterministic way.

The design with two Φ matrices only supports a single merger thread, and we believe it should cape with all n_{wt} updates coming from many threads. This is a reasonable assumption because merging at step 6 takes only about $O(|W| \cdot |T|)$ operations to execute, while ProcessBatch takes $O(n|T|I)$ operations, where n is the number of non-zero entries in the batch, I is the average number of inner iterations in ProcessBatch routine. The ratio $n/|W|$ is typically from 100 to 1000 (based on datasets in UCI Bag-Of-Words repository), and I is $10 \ldots 20$, so the ratio safely exceeds the expected number of cores (up to 32 physical CPU cores in modern workstations, and even 60 cores if we are talking about Intel Xeon Phi co-processors).

Data layout. BigARTM uses dense single-precision matrices to represent Φ and Θ. Together with the Φ matrix we store a global dictionary for all words $w \in W$. This dictionary is implemented as std::unordered_map that maps a string representation of $w \in W$ into its integer index in the Φ matrix. This dictionary can be extended automatically as more and more batches came through the

[5] http://stackoverflow.com/questions/2538070/atomic-operation-cost

system. To achieve this each batch D_b always has a local dictionary W_b, listing all words that occur in the batch. The n_{dw} elements of the batch are stored as a sparse CSR matrix (Compressed Sparse Raw format), where each row correspond to a document $d \in D_b$, and words w run over a local batch dictionary W_b.

For performance reasons Φ matrix is stored in column-major order, and Θ in row-major order. This layout ensures that $\sum_t \phi_{wt}\theta_{td}$ sum runs on sequential memory elements. In both matrices all values smaller than 10^{-16} are always replaced with zero to avoid performance issues with denormalized numbers [6].

Programming interface. All functionality of BigARTM is expressed in a set of extern C methods, To input and output complex data structures this API uses Google Protocol Buffers [7]. This approach makes it easy to integrate BigARTM into almost any research or production environment, because almost every modern language has an implementation of Google Protocol Buffers and a way of calling extern C code ("ctypes" module for Python, "loadlibrary" for Matlab, "PInvoke" for C#, etc).

On top of the extern C API BigARTM already has convenient wrappers in C++ and Python. We are also planning to implement a Java wrapper in the near future.

In addition to the programming APIs the library also also has a simple CLI interface.

Basic tools. Most of BigARTM code is written in C++, and can be built on various operating systems (Windows, Linux, and Mac OS-X) in both 32 bit and 64 bit configurations. To build the the library you need a recent C++ compiler with C++11 support (GNU GCC 4.6.3, clang 3.4 or Visual Studio 2012 or newer), and Boost C++ Libraries (version 1.46.1 or newer). All other third-parties are included in BigARTM repository:

- Google Protocol Buffers,
- Google C++ Testing Framework (gtest),
- Google Logging library for C++ (glog),
- Google Commandline flags module for C++ (gflags),
- ZeroMQ (`http://zeromq.org/`),
- rpcz (`http://code.google.com/p/rpcz/`).

We also use several free online services to host this project:

- `https://github.com/bigartm/bigartm` — to host source code,
- `https://readthedocs.org/` — free documentation hosting,
- `http://travis-ci.org` — free continuous integration service.

[6] `http://en.wikipedia.org/wiki/Denormal_number#Performance_issues`

[7] `http://code.google.com/p/protobuf/`

5 Experiments

In this section we evaluate the performance and the algorithmic quality of BigARTM against two popular software packages — Gensim [7] and Vowpal Wabbit[8]. We also demonstrate several unique BigARTM features, such as topics regularization and multi-language models, which are not available in other software packages.

All three libraries (LDA.VW, Gensim and BigARTM) work out-of-core, e.g. they are designed to process data that is too large to fit into a computer's main memory at one time. This allowed us to benchmark on a fairly large collection — 3.7 million articles from the English Wikipedia [9]. The conversion to bag-of-words was done with gensim.make_wikicorpuss script [10], which excludes all non-article pages (such as category, file, template, user pages, etc), and also pages that contain less than 50 words. The dictionary is formed by all words that occur in at lest 20 documents, but no more than in 10% documents in the collection. The resulting dictionary was caped at $|W| = 100\,000$ most frequent words.

Both Gensim and VW.LDA represents the resulting topic model as Dirichlet distribution over Φ and Θ matrices:

$$\boldsymbol{\theta}_d \sim \mathrm{Dir}(\boldsymbol{\gamma}_d), \quad \boldsymbol{\phi}_t \sim \mathrm{Dir}(\boldsymbol{\lambda}_t).$$

On contrary, BigARTM outputs a non-probabilistic matrices Φ and Θ. To compare the perplexity we will take the average of VW.LDA and Gensim distributions:

$$\boldsymbol{\theta}_d^{\mathrm{mean}} = \mathbb{E}_{\mathrm{Dir}(\boldsymbol{\gamma}_d)}\boldsymbol{\theta}_d, \quad \boldsymbol{\phi}_t^{\mathrm{mean}} = \mathbb{E}_{\mathrm{Dir}(\boldsymbol{\gamma}_d)}\boldsymbol{\phi}_t.$$

The perplexity measure is defined as

$$\mathscr{P}(D,p) = \exp\left(-\frac{1}{n}L(\Phi,\Theta)\right) = \exp\left(-\frac{1}{n}\sum_{d\in D}\sum_{w\in d} n_{dw}\ln p(w|d)\right). \qquad (6)$$

Comparison to existing software packages. The *Vowpal Wabbit (VW)* is a library of online algorithms that cover a wide range of machine learning problems. For topic modeling VW has the VW.LDA algorithm, based on the Online Variational Bayes LDA [5]. VW.LDA implementation is neither multi-core nor distributed, but an effective single-threaded implementation in C++ made it one of the fastest tools for topic modeling.

The *Gensim* library specifically targets the area of topic modeling and matrix factorization. It has two LDA implementations — LdaModel and LdaMulticore, both based on the same algorithm as VW.LDA (Online Variational Bayes LDA

[8] https://github.com/JohnLangford/vowpal_wabbit/wiki/Latent-Dirichlet-Allocation

[9] http://dumps.wikimedia.org/enwiki/20141208/enwiki-20141208-pages-articles.xml.bz2

[10] https://github.com/piskvorky/gensim/blob/develop/gensim/scripts/make_wikicorpus.py

Library	Proc.	Train Time	Inference Time	Perplexity
BigARTM Smoothing	1	62 min	127 sec	4000
Gensim LDA	1	369 min	395 sec	4161
Vowpal Wabbit LDA	1	73 min	120 sec	4108
BigARTM Smoothing	8	8 min	24 sec	4304
Gensim LDA-Multicore	8	70 min	338 sec	4470

Table 1. BigARTM comparison with LDA.VW and Gensim.LdaMulticore. Train Time is time for model training, Inference Time – time for calculation of θ_d of 100 000 held-out documents, Perplexity is calculated according to (6) on held-out documents.

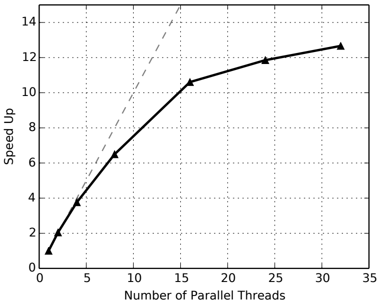

Fig. 2. BigARTM speed up

[5]). Gensim is entirely written in Python. Its high performance is achieved through the usage of NumPy library, built over a low-level BLAS libraries (such as Intel MKL, ATLAS, or OpenBLAS). In LdaModel all batches are processed sequentially, and all concurrency happens within NumPy. In LdaMulticore the workflow is similar to BigARTM — several batches are processed concurrently, and there is a single aggregation thread that asynchronously merges the results.

All runs perform one pass over the Wikipedia corpus and produce a model with $|T| = 100$ topics. The runtime is reported for an Intel-based CPU with 32 cores, $1325 MHz$. The collection was split into batches with 10000 documents each (`chunksize` in Gensim, `-minibatch` in VW.LDA). The update rule in online algorithm used $\rho = (\tau_0 + t)^{-\kappa}$, $\tau_0 = 1$, $\kappa = 0.5$, Update performed after each batch in non-parallel runs, and after P batches when running in P parallel threads. LDA priors were fixed as $\alpha = 0.1$, $\beta = 0.1$, so that $\boldsymbol{\theta}_d \sim \mathrm{Dir}(\alpha)$, $\boldsymbol{\phi}_t \sim \mathrm{Dir}(\beta)$.

Table 1 gives the comparison of runtime and perplexity between LDA.VW, Gensim and BigARTM. Chart 2 gives BigARTM speedup depending on the number CPU threads.

ToDo: discussions.

Experiments with regularization BigARTM has built-in library of regularizers that allows to improve several quality measures almost without any loss of perplexity. In the following experiment we follow the approach introduced in [11]

Model/Functional	$\mathcal{P}_{10k}$	$\mathcal{P}_{100k}$	$\mathcal{S}_{\Phi}$	$\mathcal{S}_{\Theta}$	$\mathcal{K}_s$	$\mathcal{K}_p$	$\mathcal{K}_c$
LDA	3499	3827	0.0	0.0	931	0.535	0.516
ARTM	3592	3944	96.3	80.5	1135	0.810	0.732

Table 2. Comparison of LDA and ARTM models. Quality functionals: $\mathcal{P}_{10k}$ $\mathcal{P}_{100k}$ — hold-out perplexity on 10.000 and 100.000 documents sets, $\mathcal{S}_{\Phi}$, $\mathcal{S}_{\Theta}$ — sparsity of Φ and Θ matrices (in %), $\mathcal{K}_s$, $\mathcal{K}_p$, $\mathcal{K}_c$ — average topic kernel size, purity and contrast respectively.

and configure 5% of smooth background topics and 95% sparse topical topics. In addition we enable topics decorrelation to improve kernel quality measures (kernel size, kernel purity and kernel contrast).

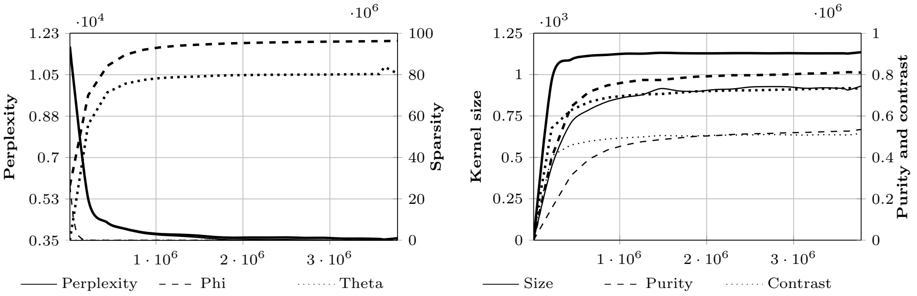

Fig. 3. Comparison of LDA (thin) and ARTM (bold) models. X axis is a number of processed documents.

Table 2 presents the comparisons results for regularized and non-regularized model. Figure 3 presents all quality measures as a function of number processed documents. Upper chart shows perplexity and sparsity of Φ, Θ matrices. Bottom chart shows average kernel measures.

ToDo: discussions.

Experiments on multi-language Wikipedia ToDo: discussions.

6 Conclusions

BigARTM architecture has a rich potential. Current components can be reused in a distributed solution that runs on cluster. Further improvement of single-node can be achieved by offloading batch processing into GPU.

Acknowledgements. The work was supported by the Russian Foundation for Basic Research grants 14-07-00847, 14-07-00908, and by Skolkovo Institute of Science and Technology (project 081-R).

References

1. Blei, D.M.: Probabilistic topic models. Communications of the ACM 55(4), 77–84 (2012)
2. Blei, D.M., Jordan, M.I.: Modeling annotated data. In: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval. pp. 127–134. ACM, New York, NY, USA (2003)
3. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet allocation. Journal of Machine Learning Research 3, 993–1022 (2003)
4. Daud, A., Li, J., Zhou, L., Muhammad, F.: Knowledge discovery through directed probabilistic topic models: a survey. Frontiers of Computer Science in China 4(2), 280–301 (2010)
5. Hoffman, M.D., Blei, D.M., Bach, F.R.: Online learning for latent dirichlet allocation. In: NIPS. pp. 856–864. Curran Associates, Inc. (2010)
6. Hofmann, T.: Probabilistic latent semantic indexing. In: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval. pp. 50–57. ACM, New York, NY, USA (1999)
7. Řehůřek, R., Sojka, P.: Software Framework for Topic Modelling with Large Corpora. In: Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks. pp. 45–50. ELRA, Valletta, Malta (May 2010), `http://is.muni.cz/publication/884893/en`
8. Rubin, T.N., Chambers, A., Smyth, P., Steyvers, M.: Statistical topic models for multi-label document classification. Machine Learning 88(1-2), 157–208 (2012)
9. Vorontsov, K.V.: Additive regularization for topic models of text collections. Doklady Mathematics 89(3), 301–304 (2014)
10. Vorontsov, K.V., Potapenko, A.A.: Additive regularization of topic models. Machine Learning, Special Issue on Data Analysis and Intelligent Optimization (2014)
11. Vorontsov, K.V., Potapenko, A.A.: Tutorial on probabilistic topic modeling: Additive regularization for stochastic matrix factorization. In: AIST'2014, Analysis of Images, Social networks and Texts. vol. 436, pp. 29–46. Springer International Publishing Switzerland, Communications in Computer and Information Science (CCIS) (2014)

Appendix

Topic t is called *regular* for the modality m if $n_{wt} + \phi_{wt}\frac{\partial R}{\partial \phi_{wt}} > 0$ for at least one term $w \in W^m$. If the reverse inequality holds for all $w \in W^m$ then topic t is called *irregular*.

Document d is called *regular* if $n_{td} + \theta_{td}\frac{\partial R}{\partial \theta_{td}} > 0$ for at least one topic $t \in T$. If the reverse inequality holds for all $t \in T$ then document d is called *irregular*.

Theorem 1. *If the function $R(\Phi, \Theta)$ is continuously differentiable and (Φ, Θ) is the local maximum of the problem* (1), (2) *then for any regular topic-modality pair (t, m) and any regular document d the system of equations* (3)–(5) *holds.*

Note 1. If a topic t is irregular then the t-th vector-column in matrix Φ^m equals zero and can not represent a discrete distribution. This means that topic t for the modality m must be excluded from the model. This mechanism is useful for irrelevant topics elimination and determining the number of topics.

Note 2. If a documents d is irregular then the d-th vector-column in matrix Θ equals zero and can not represent a discrete distribution. This means that document d must be excluded from the model. For example, a document may be too short, or have no relation to the themes of a given collection.

Proof. For the local minimum Φ^m, Θ of the problem (1), (2) the Karush–Kuhn–Tucker (KKT) conditions (see Appendix A) can be written as follows:

$$\sum_d n_{dw} \frac{\theta_{td}}{p(w \,|\, d)} + \frac{\partial R}{\partial \phi_{wt}} = \lambda_t - \lambda_{wt}; \quad \lambda_{wt} \geq 0; \quad \lambda_{wt}\phi_{wt} = 0;$$

$$\sum_m \tau_m \sum_{w \in W^m} n_{dw} \frac{\phi_{wt}}{p(w \,|\, d)} + \frac{\partial R}{\partial \theta_{td}} = \mu_d - \mu_{td}; \quad \mu_{td} \geq 0; \quad \mu_{td}\theta_{td} = 0;$$

where λ_t, λ_{wt}, μ_d, μ_{td} are KKT multipliers for normalization and nonnegativity constrains.

Let us multiply both sides of the first equation by ϕ_{wt}, both sides of the second equation by θ_{td}, and reveal the auxiliary variable p_{tdw} from (3) in the left-hand side of both equations. Then we sum the right-hand side of the first equation over d, the right-hand side of the second equation over t:

$$\phi_{wt}\lambda_t = \sum_d n_{dw} \frac{\phi_{wt}\theta_{td}}{p(w \,|\, d)} + \phi_{wt}\frac{\partial R}{\partial \phi_{wt}} = n_{wt} + \phi_{wt}\frac{\partial R}{\partial \phi_{wt}};$$

$$\theta_{td}\mu_d = \sum_m \tau_m \sum_{w \in W^m} n_{dw} \frac{\phi_{wt}\theta_{td}}{p(w \,|\, d)} + \theta_{td}\frac{\partial R}{\partial \theta_{td}} = n_{td} + \theta_{td}\frac{\partial R}{\partial \theta_{td}}.$$

An assumption that $\lambda_t \leq 0$ contradicts the regularity condition for the (t, m) pair. Then $\lambda_t > 0$. Either $\phi_{wt} = 0$ or both sides of the first equation are positive. Combining these two cases in one formula, we write:

$$\phi_{wt}\lambda_t = \max\biggl\{ n_{wt} + \phi_{wt}\frac{\partial R}{\partial \phi_{wt}}, 0 \biggr\}. \tag{7}$$

Analogously, an assumption that $\mu_d \leq 0$ contradicts the regularity condition for the document d. Then $\mu_d > 0$. Either $\theta_{td} = 0$ or both sides of the second equation are positive, consequently,

$$\theta_{td}\mu_d = \max\biggl\{ n_{td} + \theta_{td}\frac{\partial R}{\partial \theta_{td}}, 0 \biggr\}. \tag{8}$$

Let us sum both sides of the first equation over all $w \in W^m$, then both sides of the second equation over all $t \in T$:

$$\lambda_t = \sum_{w \in W^m} \max\biggl\{ n_{wt} + \phi_{wt}\frac{\partial R}{\partial \phi_{wt}}, 0 \biggr\}; \tag{9}$$

$$\mu_d = \sum_{t \in T} \max\biggl\{ n_{td} + \theta_{td}\frac{\partial R}{\partial \theta_{td}}, 0 \biggr\}. \tag{10}$$

Finally, we obtain (4) by expressing ϕ_{wt} from (7) and (9).

Analogously, we obtain (5) by expressing θ_{td} from (8) and (10). □